UNIVERSITÀ DI PARMA
Dipartimento di Ingegneria e Architettura

Public key distribution and Digital Certificates

Luca Veltri

(mail.to: luca.veltri@unipr.it)

Course of Cybersecurity, 2022/2023

http://netsec.unipr.it/veltri

Public Key Distribution

- Public key cryptography solves a major problem with symmetric algorithms
- 1) But how do you get my public key?
- 2) And how do you know it is my public key?
- In order to distribute public keys, the same scheme used for secret key distribution could be used, however
  - **the public key, used for encrypting or for verifying a signature, is not required to be secret**
    - publicly know key can be used, however
    - the key should be authenticated
- Digital certificates help to solve these two issues

Public Key Distribution (cont.)

- The simplest way is to use digital signature

$$C \rightarrow B : \quad K^+_A \text{ , } sign_{Kc^-}(K^+_A)$$

  - **B must trust C and must know the C's public key**

- In general, the signing entity may be different from the entity that B receives the key from

$$D \rightarrow B : \quad K^+_A \text{ , } sign_{Kc^-}(K^+_A)$$

  - **entity B receives from an entity D the public key of A signed by a third party C**

- In order to securely associate the K^+_A to A, and the signature to C, the identities of A and C should also be included

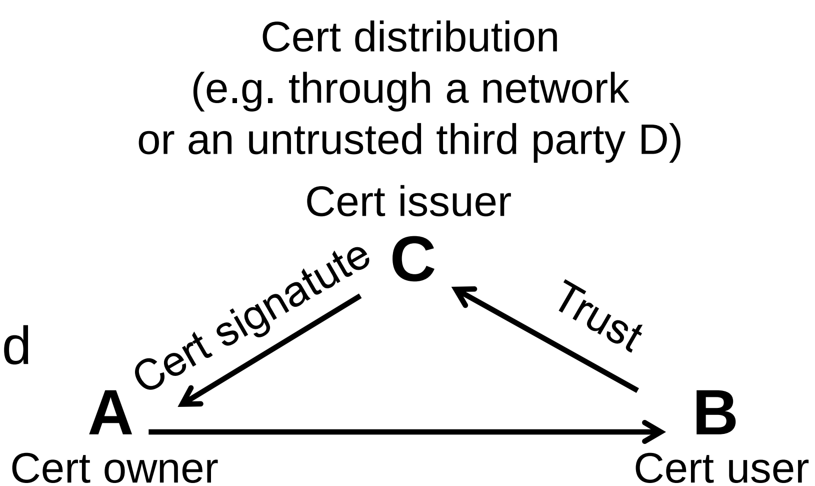
$$D \rightarrow B : \quad x = \{K^+_A || ID_A || ID_C\}, sign_{Kc^-}(X)$$

  - **the resulting data is called *digital certificate***

- In some practical cases it is directly A to send her cert to B

Digital Certificates

- Certificates are digital documents certifying the association of a public key with its owner
  - **the certification is provided through a digital signature**
    - $cert_C(A) = \{\ X=\{K_A^+, ID_A, ID_C\},\ sign_{Kc^-}(X)\ \}$
- A digital certificate may include:
  - **a public key (K_A^+)**
  - **the name of the owner of the public key (ID_A)**
  - **the name of the issuer of the certificate (ID_C)**
  - **other certificate-related information:**
    - a serial number
    - issuing and expiring date (validity)
    - other attributes
  - **the digital signature of the issuer**
    - $sign_{Kc^-}$(cert-data)
- The signature is performed by a trusted third party C
    - e.g. a Certification Authority



Digital Certificates (cont.)

- Certificates are normally used to exchange public keys in secure (authenticated) way, e.g.:
  - **for document signature**

    $A \rightarrow B$: m, $sign_{Ka^-}$(m), $cert_C$(A)
  - **for key distribution**

    $B \rightarrow A$: $cert_C$(B)

    $A \rightarrow B$: { K_S } K_B^+
  - **for entity authentication**

- Certificates are usually deployed in
  - **Web transactions**
    - HTTPS uses TLS/SSL
  - **Virtual Private Networks**
    - IPSec uses IKE
  - **Secure messaging**
    - S/MIME and PGP
  - **Anywhere strong authentication and/or encryption is required**

Digital Certificates (cont.)

- The owner of the public key (and of the certificate) could be:
  - **a person**
  - **an alias of a person**
  - **an organization**
  - **a role within an organization**
  - **a hardware system**
  - **a software system**
- Some certificates are specific (and valid only) for a subset of these entities

Certification Path

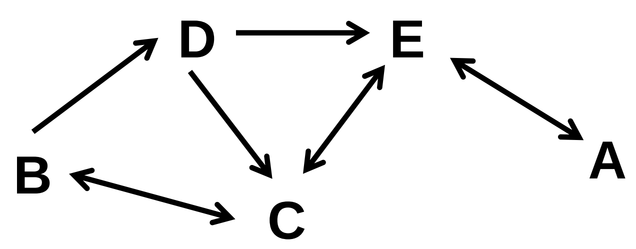
- An entity *B* requiring to use a public key of an entity *A* generally needs to obtain and validate a certificate containing the required public key
- This in turn requires that:
  - **the user *B* knows the public key of the entity C_1 that signed the certificate (that associates the identity of A to the A's pub key)**
  - **the user *B* trusts C_1**
- If the user does not already hold an assured copy of the public key of C_1, then he might use a certificate of C_1 as proof of the C_1's public key
  - **the additional certificate should be signed by a second (trusted) entity C_2 for whom B already holds an assured copy of the public key**
  - **hence, B needs to obtain two certificates: $cert_{C2}(C_1)$, $cert_{C1}(A)$**
  - **B uses the pub key of C_2 for verifying the cert of C_1; then it uses the pub key of C_1 for verifying the cert of A**

Certification Path (cont.)

- In general, a chain of certificates may be used
  - **comprising a certificate of *A* signed by one entity C_1, and zero or more additional certificates of C_i (with i=1,2,..) signed by other entities C_2, C_3, ..**
  - **$cert_{Cn}(C_{n-1})$, $cert_{Cn-1}(C_{n-2})$, .. , $cert_{C2}(C_1)$, $cert_{C1}(A)$**
- If B receives a cert chain, for verifying the cert of *A* (and all intermediate certs) he just needs the public key of the first entity C_n of the chain, or of any other C_i with $i<n$

Certification Path (cont.)

- Consider the graph G defined as follows:
  - **the vertexes are the entities**
  - **the edges represent key signatures (certificates)**
    - edge from V1 to V2 means that V1 signed the pub key of V2



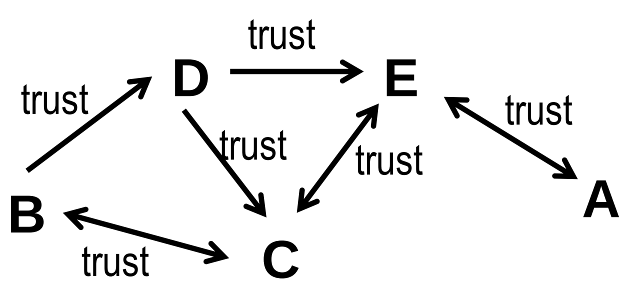
- The A's pub key can be securely distributed to B only if
  - **B knows the public key of an entity X such that there exists in G a path from X to A**
  - **the entity *B* obtains the cert chain from *X* to *A* (in order to verify the A's cert)**

Trust Path

- However, it is not sufficient to obtain the cert path
  - **_B_ needs also to trust all intermediate entities that signed the certificates included in the cert path**
    - B should trust any C_i (with $i=1,2,..,n$) to do correct use of its signature
      - i.e. signing correct certificates (connection between K^+ and its owner)
- Trust delegation (hypothesis)
  - **In case B doesn't directly know C_i, if B trusts C_{i+1}, and C_{i+1} trusts C_i, then _B_ may decide to trust C_i too**
- If this is done for each C_i (with $i=1,2,..,n$), a trust relationship from *B* to *A* (or to C_1) can be built based on the trust of *B* in C_n, and on the trust of C_i in C_{i-1} with $i=2,3, .., n$.
  - ***B* trusts C_n that trusts C_{n-1} that trusts C_{n-2}, .. , that trusts C_2 that trust C_1 (that possibly trust *A*)**
- Trust path $\mathbf{B} \xrightarrow{\text{trust}} \mathbf{C_n} \xrightarrow{\text{trust}} \mathbf{C_{n-1}} \xrightarrow{\text{trust}} \ldots \xrightarrow{\text{trust}} \mathbf{C_2} \xrightarrow{\text{trust}} \mathbf{C_1} \xrightarrow{\text{(trust)}} \mathbf{A}$

Trust Path (cont.)

- In some certificate applications, an entity signs a certificate only when he/she also trusts the certificate owner
  - **in this case, a cert path leads to a trust path**
    - the cert graph leads to a network of trust relationships
    - in some certificate applications the trust delegation is automatic (from the certificate), in some other cases it is left to the user to decide whether to trust or not a signer/certificate

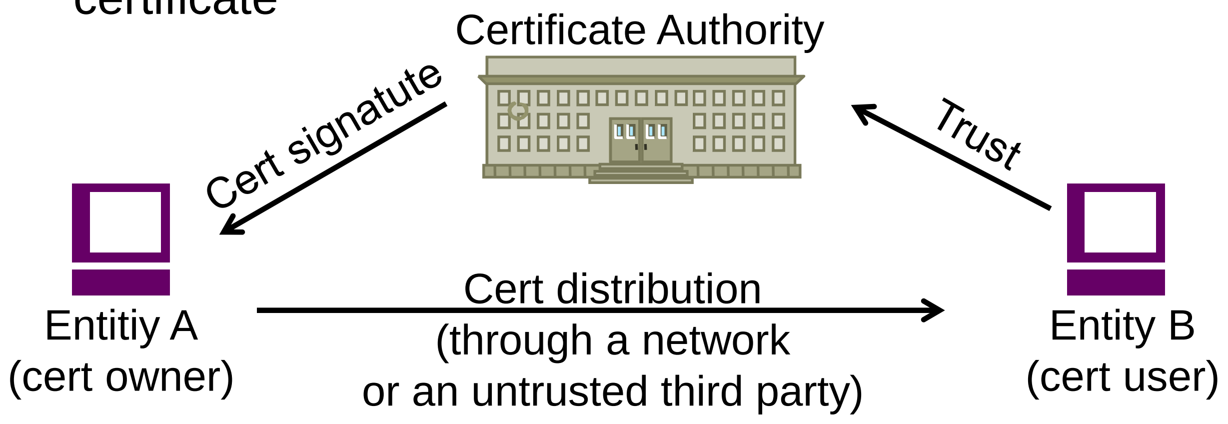


Ceritification Autorities

- In some cases, only specific (trusted) entities are allowed to sign digital certificates
  - **Certification Auhtorities (CAs)**

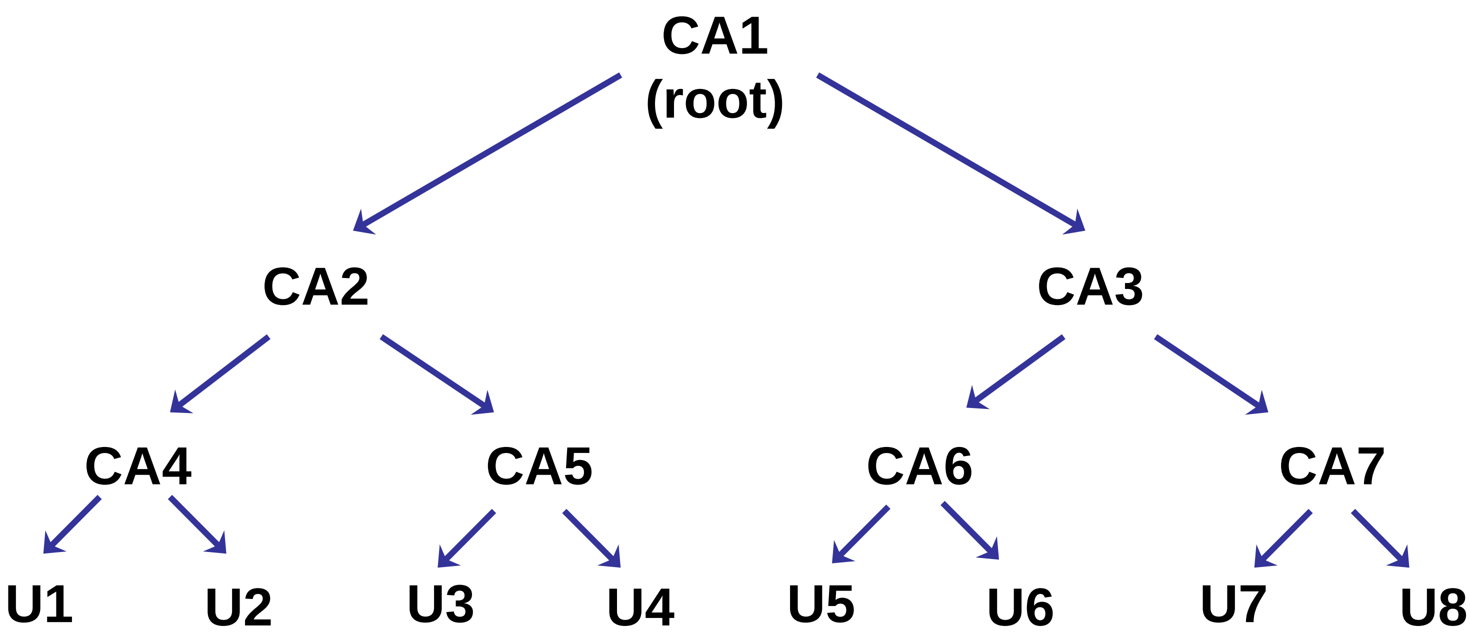
Certificate Authority

- A Certificate Authority (CA) is a trusted third party entity that issues digital certificates
  - **acts as a third party entity for certifying public keys**
    - sometimes it is the only entity entitled to do it
  - **guarantees the "connection" between public keys and their owners**
  - **trusted by the party relying upon the certificate (cert user)**
    - if a user trusts the CA and can verify the CA's signature, then she/he can also assume that a certain public key included in a certificate does indeed belong to whoever is identified in the certificate



Certificate Authority (cont.)

- CAs can be certified by other upper level CAs
- CAs may be organized in a hierarchical (trust) structure



Types of Certificates

- Depending on who signed the certificate:
  - **CA-signed certificate**
    - the certificate signature is provided by a CA
  - **User-signed certificate**
    - the certificate signature is provided by an other user
      - Note: not in X.509 standard
  - **Self-signed certificate**
    - the certificate signature is provided by the owner of the certificate

Types of Certificates (cont.)

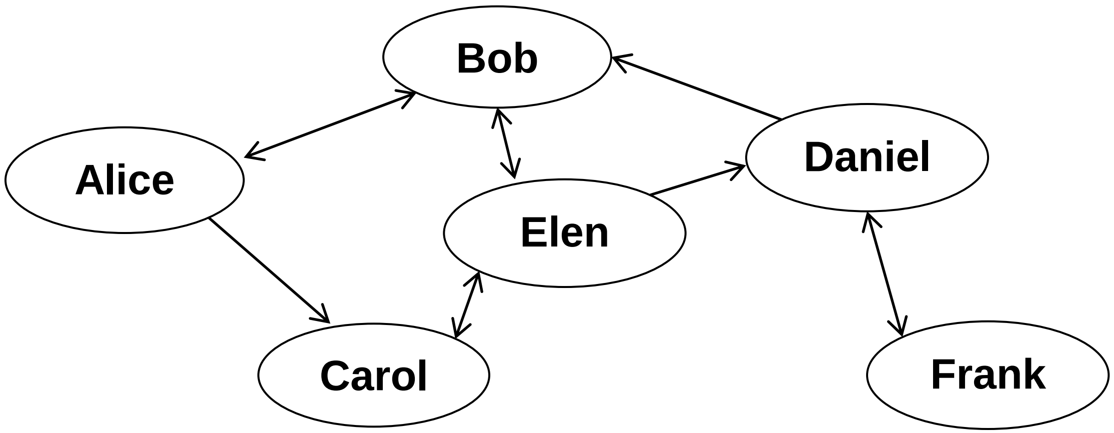
- Depending on the owner of the certificate:
  - **End system (or User) certificates**
    - for binding a user's public key to its owner
    - e.g.
      - User Certificates
        - for email or other uses
      - Server Certificates
        - for use by SSL/TLS servers
      - Software Signing Certificates
        - for signing executable code
  - **CA Certificates**
    - for verifying signatures on issued (user or CA) certificates
  - **Root Certificates**
    - self-signed by a CA

Trust models

- Public Key Infrastructure (PKI)
  - **public keys are signed only by CAs**
    - there are only CA signed certificates (and self signed CA certificates)
  - **PKI is a system for the creation, storage, and distribution of digital certificates, based on CAs (and on RAs and VAs)**
  - **it consists of:**
    - at least one Certification Authority (CA), which issues and revokes
    - certificates
    - a registration authority (RA) that acts as the verifier for the certificate authority before a digital certificate is issued to a requestor
    - a Validation Authority (VA) that can be used to validate certificates
      - e.g. publicly accessible repository (directory service) which stores Certificate Revocation Lists (CRL)
  - **centralized trust model**
  - **more CAs may be organized in a hierarchical (trust) structure**

Trust models (cont.)

- Web of Trust (WoT)
  - **decentralized trust model**
  - **users keys are digitally signed by other users**
    - i.e. user-signed certificates are used
    - the same key may be signed/certified by more than one user
      - certifying signatures
    - trust decision is left in the hands of individual users that receives such certificates



Digital certificate standards

- Widely used standards for digital certificates are X.509 and PGP
  - **ITU-T X.509 is the most common standard for digital certificates**
    - uses a PKI, i.e. hierarchical CAs
  - **PGP (Pretty Good Privacy) is mainly used for email**
    - It uses a web of trust

X.509 Digital Certificates

X.509

- ITU-T standard for a Public Key Infrastructure
- X.509 specifies, amongst other things, standard formats for:
  - **public key certificates**
  - **certificate revocation lists**
  - **attributes, and**
  - **a certification path validation algorithm**
- Some X.509 standards:
  - **IETF, RFC 5280: "Internet X.509 Public Key Infrastructure Certificate and CRL Profile"**
    - describes the X.509 v3 certificate and X.509 v2 Certificate Revocation List (CRL) for use in the Internet
  - **IETF, RFC 3647: "Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework"**
    - describes a framework to assist the writers of certificate policies or certification practice statements

X.509 History

- ITU-T X.509 (formerly CCITT X.509) or ISO/IEC/ITU 9594-8, which was first published in 1988 as part of the X.500 Directory recommendations, defines a standard certificate format
  - **The certificate format in the 1988 standard is called the version 1 (v1) format**
  - **The Internet Privacy Enhanced Mail (PEM) RFCs, published in 1993, include specifications for a PKI based on X.509 v1 certificates**
  - **When X.500 was revised in 1993, two more fields were added, resulting in the version 2 (v2) format**
  - **The experience gained in attempts to deploy PEM RFCs made it clear that the v1 and v2 certificate formats are deficient in several respects**
- ISO/IEC/ITU and ANSI X9 developed the X.509 certificate format
  - **first version in June 1996**
  - **current version 3, 2008 (RFC 5280)**
    - extends the v2 format by adding provision for additional extension fields

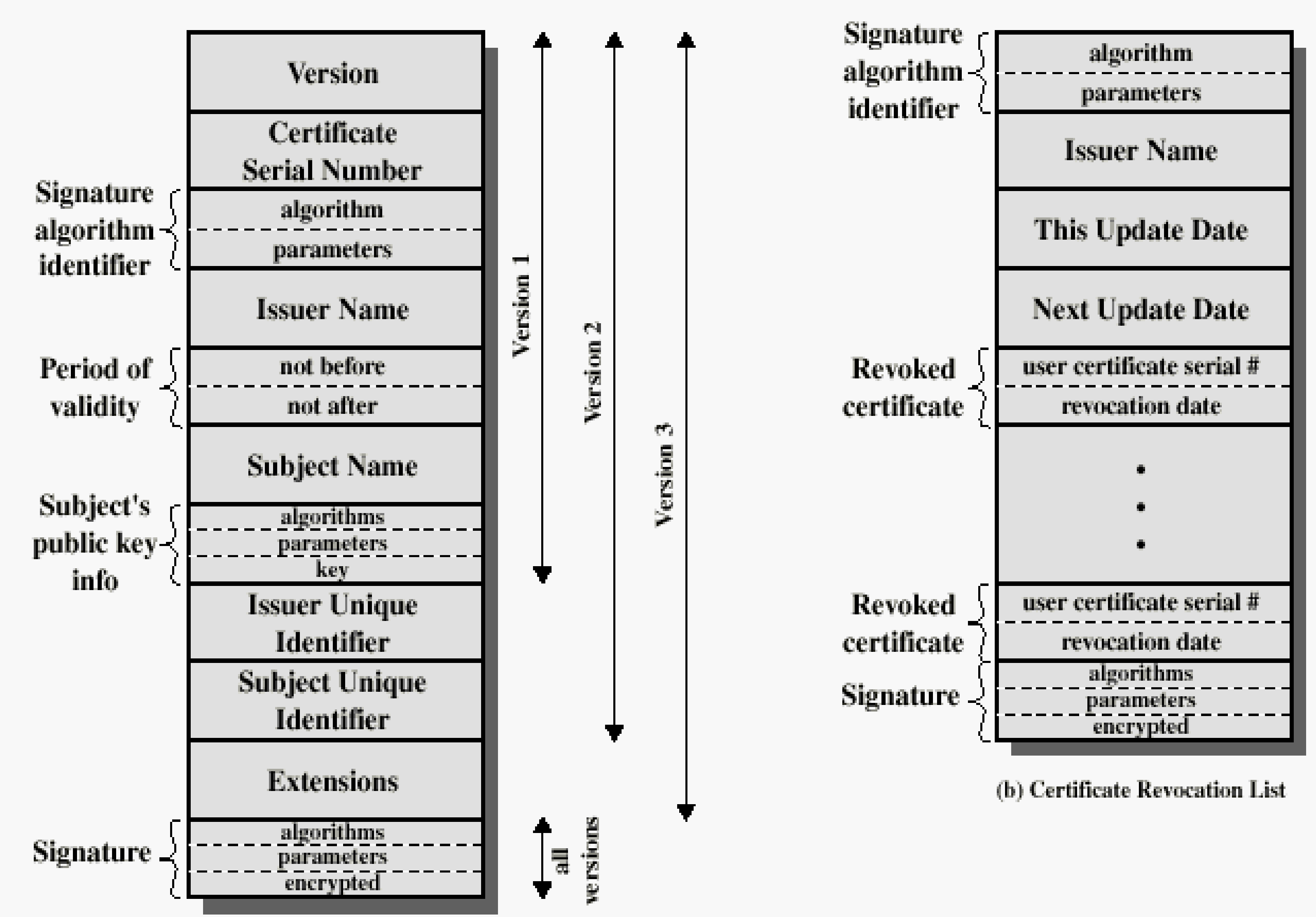
X.509 (cont.)

- The power of the X.509 model comes from its default arrangement of delegating the trust decision to CAs
- It does this by assuming trust is inherited from the signing key
- Vendors of X.509 products generally include a set of root certificates that the product will trust “out of the box”
  - **therefore automatically validate other certificates presented to the product**
  - **example**
    - X.509 encryption and signature capabilities are built into many web browsers and mail programs
      - the secure HTTP protocol (HTTPS) uses X.509

X.509 Digital Certificate

- A X.509 digital certificate includes:
  - **a public key and key info**
  - **the Distinguished Name and other information of the certificate owner that can be a person (name, e-mail, company, phone number) or system (IP address, etc.)**
  - **the Distinguished Name and other information of the certificate issuer (name, e-mail, phone number)**
  - **version number**
  - **a serial number uniquely associated to the certificate**
  - **the level of trust associated to the certificate**
  - **issue date**
  - **expiration date**
  - **digital signature of the issuer**
    - algorithm
    - signature
  - **extensions (optional)**

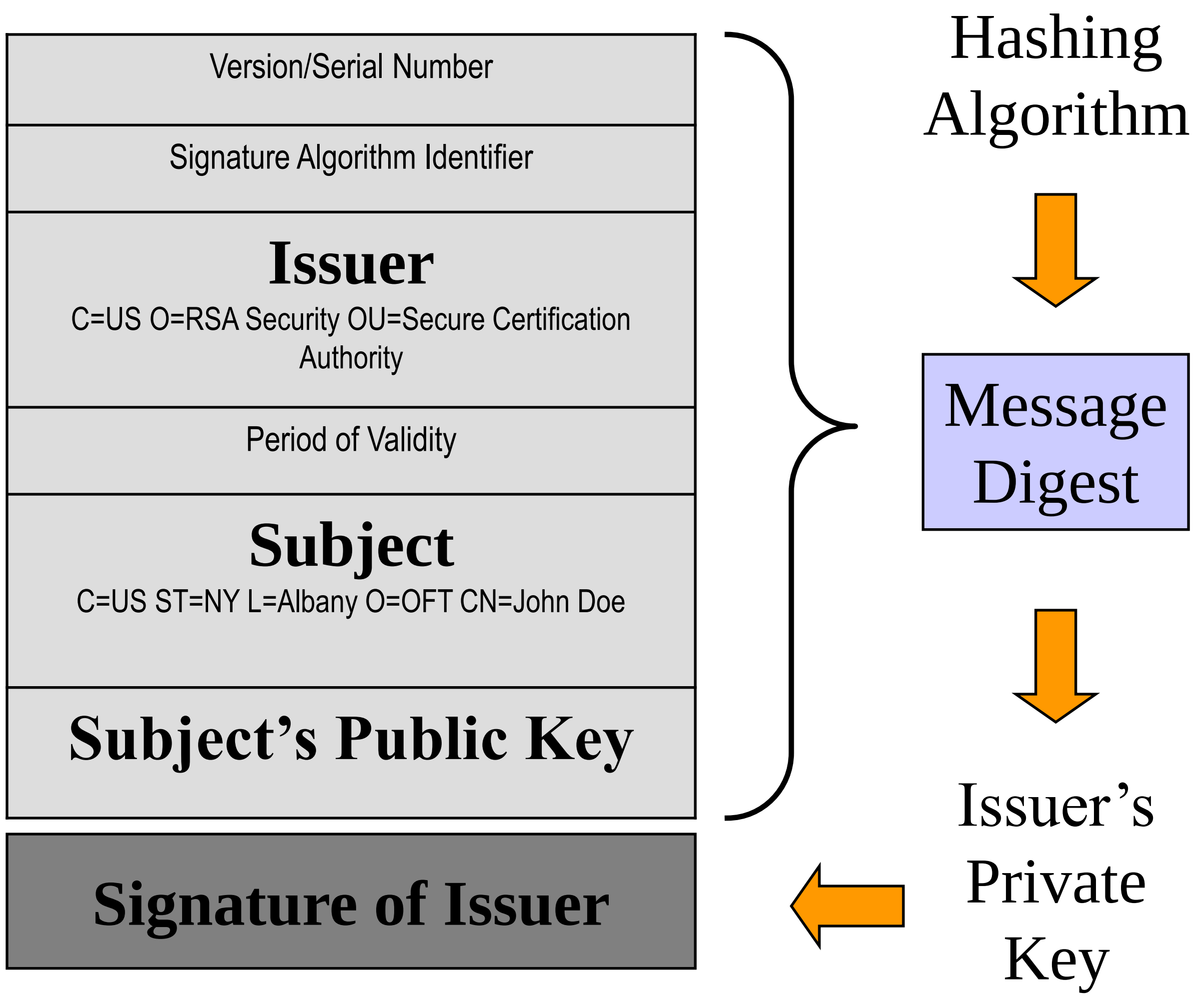
X.509 v1,v2,v3 and CRL Formats



(a) X.509 Certificate

(b) Certificate Revocation List

X.509 Certificate Signature



ASN.1, BER and DER

- ASN.1 (Abstract Syntax Notation One, defined in X.208) is the OSI's method of specifying abstract objects
- ASN.1 is a flexible notation that allows one to define a variety data types
  - **from simple types such as integers and bit strings to structured types such as sets and sequences, as well as complex types defined in terms of others**
- One set of rules for representing such objects as strings of bits is Distinguished Encoding Rules (DER), gives a unique encoding to each ASN.1 value
  - **DER is a subset of BER (Basic Encoding Rules) that describes how to represent or encode values of each ASN.1 type as a string of eight-bit octets**
    - there is generally more than one way to BER-encode a given value, while there is only one DER encoding

Example of PEM-encoded certificate


```
BEGIN CERTIFICATE
MIIC7jCCAlegAwIBAgIBATANBgkqhkiG9w0BAQQFADCBqTELMAkGA1UEBhMCWFkx
FTATBgNVBAgTDFNuYWtlIERlc2VydDETMBEGA1UEBxMKU25ha2UgVG93bjEXMBUG
A1UEChMOU25ha2UgT2lsLCBMdGQxHjAcBgNVBAsTFUNlcnRpZmljYXRlIEF1dGhv
cml0eTEVMBMGA1UEAxMMU25ha2UgT2lsIENBMR4wHAYJKoZIhvcNAQkBFg9jYUBz
bmFrZW9pbC5kb20wHhcNOTgxMDIxMDg1ODM2WhcNOTkxMDIxMDg1ODM2WjCBpzEL
MAkGA1UEBhMCWFkxFTATBgNVBAgTDFNuYWtlIERlc2VydDETMBEGA1UEBxMKU25h
a2UgVG93bjEXMBUGA1UEChMOU25ha2UgT2lsLCBMdGQxFzAVBgNVBAsTDldlYnNl
cnZlciBUZWFtMRkwFwYDVQQDExB3d3cuc25ha2VvaWwuZG9tMR8wHQYJKoZIhvcN
AQkBFhB3d3dAc25ha2VvaWwuZG9tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQDH9Ge/s2zcH+da+rPTx/DPRp3xGjHZ4GG6pCmvADIEtBtKBFAcZ64n+Dy7Np8b
vKR+yy5DGQiijsH1D/j8HlGE+q4TZ8OFk7BNBFazHxFbYI4OKMiCxdKzdif1yfaa
lWoANFlAzlSdbxeGVHoT0K+gT5w3UxwZKv2DLbCTzLZyPwIDAQABoyYwJDAPBgNV
HRMECDAGAQH/AgEAMBEGCWCGSAGG+EIBAQQEAwIAQDANBgkqhkiG9w0BAQQFAAOB
gQAZUIHAL4D09oE6Lv2k56Gp38OBDuILvwLg1v1KL8mQR+KFjghCrtpqaztZqcDt
2q2QoyulCgSzHbEGmi0EsdkPfg6mp0penssIFePYNI+/8u9HT4LuKMJX15hxBam7
dUHzICxBVC1lnHyYGjDuAMhe396lYAn8bCld1/L4NMGBCQ==
END CERTIFICATE
```

Public-key Cryptography Standard (PKCS)

- A group of public-key cryptography standards, originally developed by RSADSI (RSA Data Security Inc.)
  - **describes the syntax for messages in an abstract manner**
  - **gives complete details about algorithms**
  - **defines encoding for**
    - RSA public/private key,
    - signature,
    - short RSA-encrypted message (typically a secret key),
    - etc
- Some of these standards have moved into the "standards-track" processes of the IETF
- The standards are based on ASN.1 and BER/DER to describe and represent data

PKCS (cont.)

- PKCS #1: RSA Cryptography Standard (RFC 3447)
  - **provides recommendations for the implementation of public-key cryptography based on the RSA, covering the following aspects:**
    - Cryptographic primitives
    - Encryption schemes
    - Signature schemes
    - ASN.1 syntax for representing keys and for identifying the schemes
- PKCS #3: Diffie-Hellman Key Agreement Standard
  - **defines the Diffie–Hellman Key Agreement protocol for establishing a shared secret key between two parties**
- PKCS #5: Password-Based Cryptography Standard (RFC 2898)
  - **provides recommendations for the implementation of password-based cryptography, covering:**
    - key derivation functions
    - encryption schemes
    - message-authentication schemes

PKCS (cont.)

- PKCS #6: Extended-Certificate Syntax Standard
  - **defines extensions to the old v1 X.509 certificate specification**
  - **it has been obsoleted by the X.509v3 standard**
- PKCS #7: Cryptographic Message Syntax Standard (RFC 2315)
  - **describes a general syntax for data that may have cryptography applied to it, such as digital signatures and digital envelopes**
  - **it is compatible with PEM**
    - signed-data and signed-and-enveloped-data content can be converted into PEM messages, and vice versa
- PKCS #8: Private-Key Information Syntax Standard (RFC 5208)
  - **describes a syntax for private-key information (private key and a set of attributes**
  - **also describes a syntax for encrypted private keys**
    - a password-based encryption algorithm could be used (e.g., one of those described in PKCS#5)

PKCS (cont.)

- PKCS #9:Selected Attribute Types (RFC 2985)
  - **defines selected attribute types for use in**
    - PKCS #6 extended certificates
    - PKCS #7 digitally signed messages
    - PKCS #8 private-key information, and
    - PKCS #10 certificate-signing requests
- PKCS #10:Certification Request Syntax Standard (RFC 2986)
  - **defines a format of messages sent to a certification authority to request certification of a public key**
- PKCS #11:Cryptographic Token Interface Standard
  - **An API defining a generic interface to cryptographic tokens**
  - **Often used in single sign-on, Public-key cryptography and disk encryption systems**
- PKCS #12:Personal Information Exchange Syntax Standard
  - **defines a file format commonly used to store private keys with accompanying public key certificates, protected with a password-based symmetric key**

Certificate Revocation

- When a certificate is issued, it is expected to be in use for its entire validity period (certificates have a period of validity)
- However, various circumstances may cause a certificate to become invalid (revoked) prior to the expiration of the validity period, e.g.
  - **change of name, change of association between subject and CA (e.g., an employee terminates employment with an organization)**
  - **user's private key is assumed (or suspected) to be compromised**
  - **user is no longer certified by this CA**
  - **CA's certificate is assumed to be compromised**
- X.509 defines one method of certificate revocation
- This method involves each CA periodically issuing a signed data structure called a certificate revocation list (CRL)
- Users should check certs with CA’s CRL

Certificate Revocation List (CRL)

- A CRL is a time stamped list identifying revoked certificates which is signed by a CA and made freely available in a public repository
- When a system uses a certificate, that system not only checks the certificate signature and validity but also acquires a suitably-recent CRL and checks that the certificate serial number is not on that CRL
- A CA issues a new CRL on a regular periodic basis (e.g., hourly, daily, or weekly)
- An entry is added to the CRL as part of the next update following notification of revocation
  - **an entry may be removed from the CRL after appearing on one regularly scheduled CRL issued beyond the revoked certificate's validity period**
- CRLs may be distributed by exactly the same means as certificates themselves, namely, via untrusted communications and server systems
- One limitation of the CRL revocation method, is that the time granularity of revocation is limited to the CRL issue period